Augmented CGI Films

Mario Gutiérrez, Daniel Thalmann, Frédéric Vexo

Virtual Reality Lab (VRlab)
Swiss Federal Institute of Technology at Lausanne (EPFL)
EPFL/IC/ISIM - VRlab
INJ 130 - Station 14
CH 1015 Lausanne, Switzerland

{Mario.Gutierrez, Daniel.Thalmann, Frederic.Vexo}@epfl.ch

ABSTRACT. *The contribution of this paper focuses on the definition of a multimedia application targeted at turning CGI films into a richer and more interactive experience. We combine high-quality animated graphics to present a compelling story and enrich it by introducing a certain degree of interactivity: allowing the user to watch films using different camera paths. One of the main goals is to provide alternative ways to observe different facets of the same story, different versions according to user profiles and preferences. We propose a generic coding scheme for high-quality 3D animation, and augment it with an XML content descriptor. Based on the MPEG-7 standard, the descriptor provides information such as preset camera paths that can be freely chosen by the user.*

RÉSUMÉ. *La contribution de ce travail est focalisée sur la définition d'un système multimédia avancé qui a pour but de rendre interactif le visionnage d'un film en images de synthèse. En effet, jusqu'à lors, la seule interaction possible avec un film qu'il soit en images de synthèse ou non est de choisir un scène particulière. Notre but principal est de proposer plusieurs façons de visionner une même histoire en accord avec le profil de l'utilisateur et de ces préférences habituelles, ce qui introduit un degré certain d'interactivité tout en maintenant des coûts quasi constant de production. Pour cela, nous proposons un codage générique et de haute qualité de la géométrie de l'animation de ce monde virtuel. A cela nous rajoutons une description XML de la scène qui contient plusieurs " chemins " de caméra prédéfinis qui peuvent être choisis par l'utilisateur.*

KEYWORDS: , *CGI films, interactive multimedia applications, Computer Animation, storytelling, MPEG-7, MPEG-4, XML*

MOTS-CLÉS : *Films en images de synthèse, Application multimédia interactive, Animation par ordinateur, Narration d'Histoire, MPEG-7, MPEG-4, XML*

1. The convergence of Virtual Environments and films

This article proposes a set of coding, and descriptor schemes for a new kind of multimedia application that we call Augmented CGI (Computer Generated Imagery) Films. They are the result of blending two popular multimedia products: Virtual Environments (VE) and Computer Generated films. VE are interesting due to the experience they create by mixing impressive graphics, sound, textual information and interaction. While VE provide the most interactive experiences, they usually fail to present rich stories. Complex narratives in video games are sacrificed for the sake of interaction. The contrary occurs with CGI films. Also known as Computer Animation (CA) films, they immerse the spectator in a compelling multimedia experience by means of impressive special effects, showing images and shots that would be impossible to make in the "real world". CA films concentrate on conveying a message, a narrative with different degrees of complexity, but leave interaction aside.

Researchers are studying the tension between author and user control of narratives in VE and multimedia systems such as films. In [STE 04] a system is proposed to better balance user freedom and author control. The authors chose to stay on the side of VE. For them, keeping enough interactivity -free exploration of the virtual world- is key to achieve immersion and engagement. However, they report that despite the narrative event adaptation, the level of user comprehension is still higher in non-interactive multimedia systems (movies) than in fully interactive VE.

We have chosen the side of CGI movies and decided to introduce certain levels of interaction and additional information. Our goal is to give the user alternative ways to interpret the story without sacrificing neither the narrative comprehension nor the story. Figure 1 shows where we situate our proposal compared to video games and CGI films. The main objective of a film is to tell a particular interpretation of a story, the result of the director's selection of scenes and camera paths. Giving the spectator the option to select from multiple camera paths could be an interesting added value. It could also add new expressivity dimensions to this art. Scenes could be presented in different ways, adapted to different user profiles (age, culture, preferences etc.). Providing multiple camera paths was originally proposed for DVD films. However it is rare to have this option due to the high costs of filming a live scene using many cameras simultaneously. The space available on the media is also an issue: DVD's are able to store only about 2 hours of video, only one camera path. Having multiple camera paths seems to be the natural next step for CGI films. We call them 3D animation movies, but we only watch them as 2D images. The coding and film descriptors we propose provide a good foundation to enrich -augment- the expressivity and interactivity of CGI films. Of particular interest is the notion of profiling based on the camera paths. The same animation could be rated for a general public or a restricted audience according to the camera path being used. E.g. violent scenes could be shown using a camera path that hides the most aggressive images.

Creating augmented CGI films requires solving several problems. We must define an efficient coding scheme to represent high-quality animated 3D geometry. Films

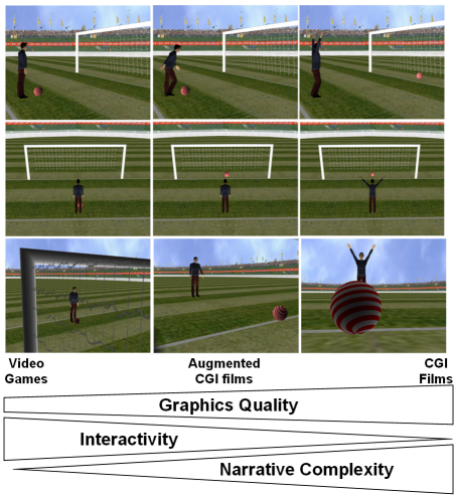

Figure 1. *Augmented CGI films are in the middle of the multimedia apps. spectrum.*

must be compact and simple enough to be decoded on a user terminal in real-time. Target platforms would be digital set-top boxes [O'D 99], powered by chips with limited performance and capacities (no Floating Point Unit). For the coding scheme to be truly useful, the encoding tools should be seamlessly integrated into the current production tools. Making a CGI movie is a complex and long process involving many design, production and post-production steps. We consider a production pipeline centered on a single modeling and animation tool. The movies we are able to encode are animated scenes created using Alias Maya as the main tool. Next section presents an overview of the state of the art on coding and compression of animated 3D geometry. Then we describe our coding/compression scheme. We continue by addressing the problem of defining a film descriptor and a user interface. After we describe the implemented encoder and decoder prototypes.

2. Coding of 3D animation

When reviewing the state of the art on 3D animation coding, it is natural to be pointed towards compression techniques.

Our search for efficient coding schemes led us to the MPEG-4 specification [ISO]. MPEG-4 provides very low-bit rate coding for virtual characters and efficient compression for general 3D shapes [CAP 00]. However, the MPEG-4 coding scheme does not support advanced animation techniques, neither for virtual characters nor for general 3D objects. This forces designers to produce low visual quality contents - created with the limited number of supported animation techniques. Efforts have been

done to incorporate advanced animation techniques to the specification, but they have mainly focused on the animation of virtual characters, e.g. Bone Based Animation [PRE 02] or morphing. Furthermore, the MPEG-4 specification has grown too much to be implemented in a lightweight terminal (decoder) and the same applies to the encoder. The Moving Picture Experts Group has recently identified the need for a lightweight representation for interactive scenes [Duf], [Mov]. While focusing on mobile applications and mainly 2D content, this kind of initiatives show the need for a generic and simple representation for animated 3D content.

Research concerning 3D geometry coding and compression includes the work of Rossignac, the Edgebreaker algorithm [ROS 01]. Focused on polygonal meshes compression, this algorithm is less complex and easier to implement in a lightweight terminal than previous compression algorithms. A detailed report on 3D compression algorithms can be found in [TAU]. Nevertheless, this research concentrates on static geometry compression.

Lengyel was one of the first to consider time-dependent geometry as a streaming media and proposed a technique for compressing data streams of dynamic meshes [LEN 99]. His algorithm is based on compressing the motion of the vertices of the animated meshes by means of a predefined set of fitting predictors. Lengyel affirms that in most of the cases the best possible compression would be to encode the modeling primitives used to create the animation. However, he concludes that it would be unfeasible to implement every single primitive on the run-time engine -user terminal- since this component must remain as generic and fast as possible.

The Dynapack algorithm [IBA 03] performs a space-time compression of animated triangle meshes with fixed connectivity by means of a single predictor for all of the vertices and for all key frames. This approach differs from Lengyel's algorithm in the sense that it avoids the need for grouping vertices to be animated with different predictors -the ones that best fit the motion of each group.

Applying a single generic method -single predictor- for decoding the animation simplifies the implementation of the user terminal. In this sense, the Geometry Videos technique proposed by Briceño et. al. [BRI 03] goes one step further. Not only it decodes the animation by applying a generic algorithm, but it takes advantage of existing mature video processing and compression techniques to increase the compression ratio. However, the technique is viable only for fixed-connectivity meshes and certain types of animation.

Karni and Gotsman [KAR 04] defined a compression scheme based on the principle component analysis (PCA) method. They represented the animation sequence (spatial correlation) using a small number of basis functions. They exploit temporal correlations to increase the compression rate by using second-order linear prediction coding (LPC). Their algorithm achieves higher compression rates than other algorithms but the encoding time is longer, unsuitable for real-time applications.

The works cited before were focused on maximizing the data compression, the bit per vertex rate. These algorithms have been tested in a research context situated at

some distance from the film production pipelines. Problems such as how to integrate the compression techniques into the production tools and user context are not considered. Moreover, the best techniques are usually lossy compression algorithms, they can degrade the quality of the animation and hence do not fulfill our requirements.

The compression and coding scheme we are looking for should keep an equilibrium between the following factors: fast decoding (acceptable frame rate on a user terminal: at least 25 fps), adaptability to the digital content production pipeline. The coding scheme should be generic enough to be integrated into modeling/animation tools and without imposing restrictions to the designers. Creating an animated sequence requires applying many different animation techniques. The encoding tool should seamlessly integrate into the production stage. Our proposal does not focus on obtaining high compression rates but on defining a simple generic representation.

3. A coding scheme for augmented CGI films

The representation for 3D films that we defined allows for transparent integration into production tools and can be implemented in low-end hardware since it does not require floating point operations. This algorithm does not impose any limitation to the animation techniques used by the digital artist: physics-based animation, morphing, bone-based, etc. We work with the final version of the animation and don't require to know how it was produced. The coding algorithm focuses on the faithful reproduction (loss-less coding) of a broad range of animation effects created with state of the art modeling tools such as Maya.

As mentioned in the introduction, we tried to minimize the floating point operations required to decode the 3D film. This prevented us from using more efficient compression algorithms such as the ones cited in section 2. Moreover, high-compression-rate algorithms tend to sacrifice the versatility of the encoder: they reduce the variety of applicable animation techniques -see the case of MPEG-4 in section 2- and/or impose relatively high hardware requirements. The 3D animation coding scheme we propose goes in a direction similar to the call for proposals from MPEG-4 for a light-weight scene representation [Mov].

```
CGI_film{
 string  filmName
 unsigned int framesInFilm
 unsigned int frameRate
 CGI_scene scenes[ ]
}

CGI_scene{
 unsigned int initFrame
 unsigned int endFrame
 CGI_object objectsInScene[ ]
}
```

```
CGI_object{
  string  objectName
  quantized_float   vertices[ ]
  unsigned int coordIndex[ ]
  quantized_float   textureCoordinates[ ]
  CGI_texturedShape texturedShapes[ ]
}

CGI_texturedShape{
  string  textureImageURL
  unsigned int texCoordIndex[ ]
}
```

Figure 2. *Coding syntax for the geometry of augmented CGI films.*

3.1. *Geometry coding*

The geometric representation we have adopted is based on the indexed face set format -polygonal mesh- and contains a list of points in 3D or 2D, an array defining the mesh, the texture file to map and the uv coordinates. The format resembles a simplified VRML file. Objects are defined in global coordinates. All transformations -translation, rotation, vertex displacements, etc.- are defined by transition functions -polynomial curves. The data structure for the geometry is defined in figure 2.

CGI_film is the main geometry container, it provides basic information such as the film's name, the number of animation frames, the frame rate and a pointer to an array of **CGI_scene** nodes. **CGI_scene** is the basic building block of an augmented CGI film. It defines the 3D objects present during a certain frames interval. Dividing the film into scenes -or chapters- will allow us to associate MPEG-7 descriptors which will provide information on a per-chapter basis. The chapter descriptors will contain text descriptions and in particular the available camera paths to watch the scene using different camera paths, see section 3.3.

CGI_object defines the 3D objects that participate in a scene, whether they are animated or not. CGI_objects are constituted of a single mesh, without skeleton or any other hierarchic structure. **vertices** is a list of quantized_floats (integer values used to represent floats) specifying the vertex coordinates (3D vectors). **coordIndex** is a list of integers containing the indices to the vertices describing the facets of the object's mesh, -1 is used as separation character, e.g.: 0,1,2-1, 2,1,3,-1 describes two triangles. **textureCoordinates** is a list of uv coordinates used to map the texture over the object's mesh. Several textures can be mapped over different mesh sections of an object. We "bake down" the output of the rendered texture, so that each single texture has every rendered feature: shadows, scattered light, illuminance, transparency etc. Textures can be updated during animation to reproduce lighting effects such as shadows without need to calculate them on client side. **texturedShapes** is a list of **CGI_texturedShape** nodes containing the URL pointing to the image texture and the indices to the **textureCoordinate** list used to set the uv coordinates on each vertex.

```
CGI_animation{
  string objectName
  nibble  contentToUpdate

  CGI_transitionTexture textureUpdates[ ]

  unsigned int    initFrame
  unsigned int    endFrame
  CGI_transitionFunction  transFunctions[ ]
}
```

```
CGI_transitionFunction{
  nibble  component
  unsigned int  polynomDegree
  quantized_float  polynomialCurve[ ]
  unsigned int   vertexCluster[ ]
}

CGI_transitionTexture{
  unsigned int textureIndex
  CGI_texturedShape textureShape
}
```

Figure 3. *Vertex-based animation coding syntax.*

3.2. *Animation coding*

The trajectory of each vertex in the scene is coded using polynomial curves per vertex component. The algorithm first tries to fit to a polynomial curve the trajectory of each vertex component. Our tests showed parabolic interpolation allows for coding in a compact way the trajectories -transition functions- of the vertices being animated.

Thus, after the first step we have a set of parabolic equations (2nd degree polynomials) describing the trajectory of each vertex component for every vertex in the animation. Each transition function describes a vertex component trajectory for at least 3 animation frames (3 points needed to calculate a parabolic equation).

Some Vertices do not require transition functions, others are described using only lines (first degree polynomials). In many cases a single parabolic equation is able to describe a vertex component trajectory for more than 4 animation frames. This depends on the nature of the animation.

In order to increase the compression ratio, vertices with the same transition function are grouped into clusters. This is done on a second step where we analyze the whole set of transition functions and group vertex components into clusters animated by the same function. This is illustrated in figure 4.

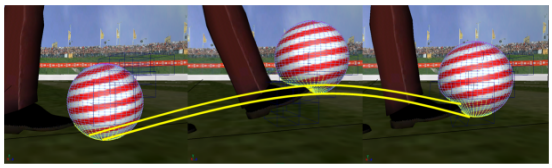

Figure 4. *Vertices follow parabolic trajectories that can be grouped into clusters.*

The animation encoding includes also the rendering of the surface textures in order to "bake-down" the lighting effects. At each animation frame, texture images are compared to the previous ones and if they have differences over a certain epsilon, a new texture update is registered together with the corresponding image. Image comparison is done on a per-pixel basis, two texture images are significantly different if they have a certain percentage of different pixels. Two pixels are different if the euclidean distance between them surpasses a certain threshold (pixels are represented by 3 coordinates: RGB components). The syntax used to represent the animation functions and texture updates is described in figure 3.

CGI_animation is the basic update message, it indicates the object to which the current update is to be applied. **contentToUpdate** is a flag that indicates whether the update contains a set of transition functions and/or texture updates for the current object. **textureUpdates** is a list of **CGI_transitionTexture** nodes which indicate the index to the texture that will be updated with the new **textureShape** node. **initFrame** and **endFrame** define the interval where the list of transition functions

(**transFunctions**) will be applied. **CGI_transitionFunction** indicates the list of vertices (**vertexCluster**) to be animated by evaluating the polynomial defined in **polynomialCurve** (list of **polynomDegree** + 1 coefficients).

3.3. *Coding camera paths through MPEG-7-based descriptors*

MPEG-7 [SAL 02] has been selected as the best way to describe the "augmented" features (different camera paths) of our coding scheme. The information we will associate to the animation and geometry data is the following: a text description of the film and each of its chapters (CGI_scenes which are the building blocks of the film, see section 3.2), plus the available camera paths per scene.

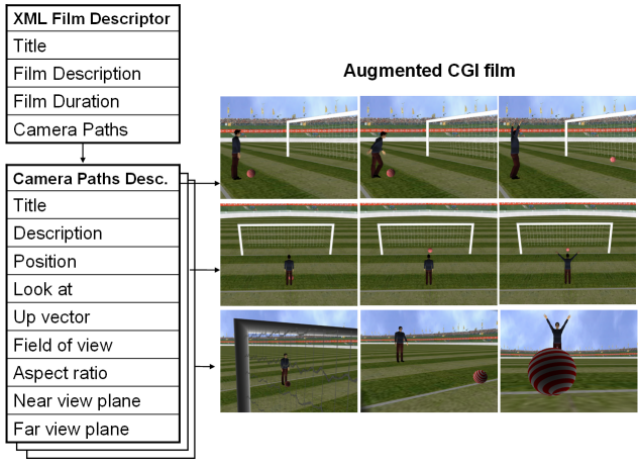

Figure 5. *XML-based descriptors for Augmented CGI films.*

A new descriptor for the camera paths was needed. The ones defined on the Visual [YAM] and the Multimedia Description Schemes [BEE] parts of the standard were created to describe 2D video. We did not find them suitable for specifying camera paths in 3D. The parameters needed to define a camera view on a 3D application are those used to create the view (eye point, look-at point, "up" direction) and projection matrices (field of view, aspect ratio, near and far view-planes) [Mic]. They cannot be mapped to the camera motion parameters defined in the MPEG-7 CameraMotion Descriptor [YAM], which defines basic camera operations such as: fixed, tracking booming, dollying, panning, tilting, rolling and zooming.

Each scene in an augmented CGI film has a short text description associated as well as the definition of the available camera paths. MPEG-7 provides descriptors to incorporate semantics (text descriptions) into video sequences, we will use them for our animation sequences. The new descriptor we propose will be used to describe

the camera animation, including the parameters required to construct the view and projection matrices.

To describe the path and animation sequence of a camera, we keep track of its position/orientation (view matrix parameters), and projection matrix parameters such as the field of view, etc. We follow the same approach used to encode the geometry animation and describe the camera animation through polynomial curves. Modeling/Animation tools describe camera paths by means of spline curves which are easy to handle by designers and provide a compact representation. We avoid using this representation to keep the decoder simple and minimize the floating point operations. Evaluating a polynomial is less expensive than interpolating a spline curve. Moreover, the calculation routine has been already implemented in the decoder to reproduce the animation. Figure 5 shows a schematic view of the descriptors.

4. User interface for augmented CGI films

The previous section presented the coding and description representations we have defined to create augmented CGI films. We must now consider the problem of controlling and interacting with such content. Since we are talking about films, the most natural interface would be something similar to the classical remote control used to interact with most of the home electronic devices such as TV or DVD players. The majority of remote controls rely on text displayed on the TV screen and/or small displays embedded in the devices (e.g. VCR display), but not on the remote control. We have decided to apply a user interface we have recently developed in the context of Virtual Environments interaction: the mobile animator [GUT 04]. This is a PDA-based interface that gives the user full control of the virtual objects in the VE. The PDA interface eliminates the need for overlaying menus or other widgets that obstruct the main display screen. Based on this technology, we extend the paradigm of the remote control and provide an interface that allows for selecting the camera path to apply. The text descriptions associated to the film (title, duration, etc.) and to each camera view, if any, are presented as well on the PDA display (see figure 6).

5. Implementation and results

A prototype has been implemented in order to test the coding scheme and descriptor we are proposing. An overview of the main components we have implemented is presented in figure 7. With this prototype we tried to cover the whole digital content life cycle, starting with the content production and finishing with the augmented CGI film. We developed a Maya plug-in that exports geometry, animation and camera descriptors to the defined formats. The plug-in works with virtually any geometry -NURBS or polygonal meshes- and is independent of the techniques used to create the animation. It was developed in C++ using the Maya API. The film viewer is a DirectX-based application developed in C++. The XML descriptors are parsed using the Xerces C++ library [Apa]. The viewer uses TCP-based sockets to send and re-

Figure 6. *User interface for Augmented CGI films.*

ceive data to/from the PDA interface. The PDA runs an application developed using embedded VisualBasic and it does not handle the full XML descriptor but a simplified version containing only the list of available camera views and associated text.

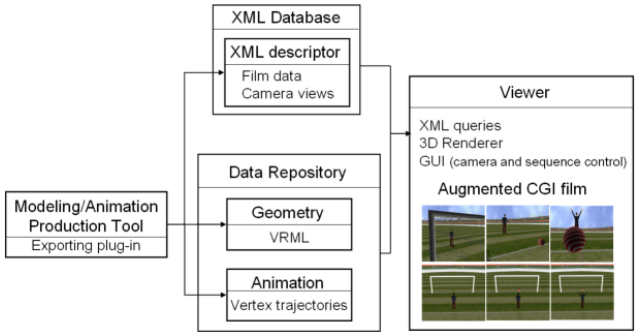

Figure 7. *Main components of the implemented prototype.*

We have done tests with an animated scene where a virtual character scores a goal in a football stadium, see figure 6. The virtual character is animated through a skeleton similar to the one defined by the H-Anim standard [HAN] and uses smooth skinning for calculating mesh deformations. Three cameras have been defined (side, top views and a mobile camera). Table 1 presents some statistics showing that the coding scheme can be stored in a DVD media and thus be distributed as a conventional film: 160 animation frames at 30 frames per second makes 5.3 seconds of animation and requires 6340KB of data (geometry + animation + camera paths). We obtain an average of 5GB of data for a 1-hour augmented CGI film. According to our estimations, a two-hour augmented CGI film can be stored in a double-sided DVD. Being able to distribute the content in a standard format is important if we consider the perspectives of this technique as a commercial product.

File Type	Binary File Size
Original scene in Maya format (24000 polygons, 160 animation frames)	1561KB
Geometry	290KB
Animation	6000KB
Camera Paths	50KB
1 second of "augmented" animation	1300KB

Table 1. *Statistics for the Augmented CGI films coding scheme.*

6. Conclusions

We have presented the first version of a coding scheme and MPEG-7-based (XML) descriptor for 3D animated scenes that are augmented through the incorporation of alternative viewpoints customizable by the user, as well as additional information about the scene. Augmented CGI films are the result of blending Virtual Environments and Computer Generated films. We introduced some level of interactivity, a typical VE feature, while keeping high quality animated graphics. We believe this gives content producers new expression possibilities to tell a story and create multimedia experiences. The results we have presented show the feasibility of producing augmented CGI films using commercial modeling tools such as Maya and implementing a client application with a user-friendly interface. The vertex-based animation format does not require implementing any special algorithm on the client side in order to faithfully reproduce the animation. Virtually any combination of animation techniques can be used to animate the scene. Our technique guaranties that the animation will be accurately reproduced. The evaluation of polynomial curves is trivial and allows for implementing light-weight clients (viewer applications). Sound handling and the use of 3D sound effects would enhance the experience of viewing films from different angles. This is let as future work.

7. References

[Apa] APACHE SOFTWARE FOUNDATION., "Xerces-C++, validating XML parser. http://xml.apache.org/xerces-c/".

[BEE] VAN BEEK P., BENITEZ A. B., HEUER J., MARTINEZ J., SALEMBIER P., SHIBATA Y., SMITH J. R., WALKER T., "15938-5/FCD Information Technology, Multimedia Content Description Interface, Part 5 Multimedia Description Schemes, Tech. rep. N3966, ISO/IEC JTC1/SC29/WG11, Singapore, SG, March 2001".

[BRI 03] BRICEÑO H. M., SANDER P. V., MCMILLAN L., GORTLER S., HOPPE H., "Geometry videos: a new representation for 3D animations", *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, 2003, p. 136–146.

[CAP 00] CAPIN T., PETAJAN E., OSTERMANN J., "Very Low Bitrate coding of virtual human animation in MPEG-4", *IEEE International Conference on Multimedia and Expo (ICME)*, vol. 2, 2000.

[Duf] DUFOURD J-C. (EDITOR), "Rationale and Draft Requirements for a Simple Scene Description Format, October 2003. ISO/IEC JTC1/SC29/WG11 Document No. N6038.".

[GUT 04] GUTIERREZ M., VEXO F., THALMANN D., "The Mobile Animator: Interactive Character Animation in Collaborative Virtual Environments", *IEEE Virtual Reality 2004 (to appear)*, Chicago, USA, March 2004, p. 125–132.

[HAN] H-ANIM, "The humanoid animation working group. http://www.h-anim.org".

[IBA 03] IBARRIA L., ROSSIGNAC J., "Dynapack: space-time compression of the 3D animations of triangle meshes with fixed connectivity", *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, 2003, p. 126–135.

[ISO] ISO/IEC 14496-2:1999, "Information Technology – Coding of Audio-Visual Objects, Part 1: Systems (MPEG-4 v.2), December 1999. ISO/IEC JTC 1/SC 29/WG 11 Document No. W2739.".

[KAR 04] KARNI Z., GOTSMAN C., "Compression of soft-body animation sequences", *Computers & Graphics*, vol. 28, num. 1, 2004, p. 25–34.

[LEN 99] LENGYEL J. E., "Compression of time-dependent geometry", *Proceedings of the 1999 symposium on Interactive 3D graphics*, ACM Press, 1999, p. 89–95.

[Mic] MICROSOFT DEVELOPER NETWORK, "Using Matrices, DirectX 9 Tutorials, Microsoft Corporation 2004. http://msdn.microsoft.com".

[Mov] MOVING PICTURE EXPERTS GROUP, "Call for Proposals for Lightweight Scene Representation, Munchen – March 2004. ISO/IEC JTC1/SC29/WG11 Document No. N6337.".

[O'D 99] O'DRISCOLL G., *The Essential Guide to Digital Set-Top Boxes and Interactive TV*, Prentice Hall PTR, 1999.

[PRE 02] PREDA M., PRÊTEUX F., "Advanced animation framework for virtual character within the MPEG-4 standard", *Proceedings IEEE International Conference on Image Processing (ICIP'2002), Rochester, NY*, vol. 3, 22-25 September 2002, p. 509-512.

[ROS 01] ROSSIGNAC J., "3D compression made simple: Edgebreaker with ZipandWrap on a corner-table", *Proceedings of the SMI 2001 International Conference on Shape Modeling and Applications*, 2001, p. 278–283.

[SAL 02] SALEMBIER P., "Overview of the MPEG-7 Standard and of Future Challenges for Visual Information Analysis", *EURASIP Journal on Applied Signal Processing*, vol. 4, 2002, p. 1–11, Hindawi Publishing Corporation.

[STE 04] STEINER K. E., TOMKINS J., "Narrative event adaptation in virtual environments", *Proceedings of the 9th international conference on Intelligent user interface*, ACM Press, 2004, p. 46–53.

[TAU] TAUBIN G., "3D Geometry Compression and Progressive Transmission, Eurographics State of the Art Report, September 1999.".

[YAM] YAMADA A., PICKERING M., JEANNIN S., CIEPLINSKI L., OHM J.-R., KIM M., "15938-3/FCD Information Technology, Multimedia Content Description Interface, Part 3 Visual, Tech. Rep. N4062, ISO/IEC JTC1/SC29/WG11, Singapore, SG, March 2001".